Do It Yourself: Generic Pseudo-labeling for Domain Adaptation in Deep Image Classification

Tsirmpas Dimitris

30th May 2024

1 Introduction

In the domain of deep learning, the quest for developing models that can leverage enormous sources of unlabeled data, in addition to limited labeled data, has spawned different paradigms for semi-supervised learning. In this regard, pseudo-labeling has emerged as one of the leading paradigms, promising a path toward bridging the chasm between supervised and unsupervised learning because of its simplicity, effectiveness, and adaptability across various tasks, especially computer vision tasks such as image classification.

The core of pseudo-labeling lies in the principle of iterative refinement of model predictions over unlabeled data. Given a model trained on a small labeled dataset, predictions are made on the unlabeled data. Those predictions are used to assign pseudo-labels to the unlabeled instances, effectively enlarging the labeled dataset. Further, the model is trained again on the augmented dataset in an iterative fashion, refining its performance and pseudo-labeling accuracy. It thus aims to gradually align the feature distributions of the source and target domains.

The efficiency of pseudo-labeling is conditioned on several key factors: the quality of the initially labeled data, the heuristics of pseudo-labeling, and the model architecture. These conditions have led to the inception of several algorithms, tightly coupled with specialized models able to learn domain-invariant features from both datasets. However, the level of complexity and cost to these models can be daunting, with most recent approaches necessitating multiple Deep Neural Network models (DNNs) as well as a suite of additional algorithms and training frameworks.

In this paper we propose a simple, alternative approach to training models on both unsupervised and semi-supervised datasets. Our approach seeks to minimize the training cost, only necessitating the presence of the original model, and keeping additional tuning hyper-parameters to a minimum. We evaluate the performance of our algorithm on both a small and large dataset, and extract useful insights on the weaknesses of pseudo-labeled data in the absence of the aforementioned specialized algorithms.

2 Related Works

Pseudo-labeling was first introduced to DNNs by [7]. The assumption behind the procedure is that data points are clustered according to their class, which the discriminator needs to find the border of. By using pseudo-labeling, the discriminator can refine the border regions of these clusters by gradually including data points further and further from the cluster's center. Thus, pseudo-labeling can be seen as a form of Entropy Regularization [4].

One of the ways where pseudo-labeling fails was explored by [8]. As demonstrated in the "twins-moons" dataset, should the initial clusters not be evaluated correctly at the beginning of the algorithm, the discriminator could learn an entirely wrong decision boundary. A classic example of this happening would be very few initial samples being used during training. In the context of domain adaptation however, it could also be caused by a sufficiently large gap between the source and target distributions.

The latter issue is the primary reason for the current gallery of additional models and algorithms involved in the main training loop of SOTA domain adaptation systems using pseudo-labeling. [2] outline most of the most commonly used algorithms, including model regularization, outlier filtering (using the cluster assumption introduced by [7]) and domain alignment. [12] showcase modern SOTA models from different domain-adaptation approaches, as well as a review of common datasets.

In the original pseudo-labeling paper, [7] proposed a training loop where both the source and target distributions were considered at different points in time using a weighted sum of losses from both datasets. Specifically,

$$\mathcal{L} = \mathcal{L}_{source} + a(t) * \mathcal{L}_{target}, \text{ where } \alpha(t) = \begin{cases} 0 & \text{if } t < T_1 \\ \frac{t-T_1}{T_2-T_1}\alpha_f & \text{if } T_1 \le t \le T_2 \\ \alpha_f & \text{if } T_2 < t \end{cases} \text{ where } T_1, T_2, \alpha_f \text{ hyperparameters.}$$

This approach however requires the definition of multiple hyperparameters, requiring the existence of an extensive target validation set, which is usually hard to construct, given the problem formulation of domain adaptation. [11] propose a different method of scheduling the source and target dataset losses, which we elaborate on later in this paper.

Finally, [8] denote a number of important steps towards correctly evaluating pseudo-labeling algorithms. Notably, they stress the importance of not over-tweaking hyperparameters since, as stated above, large validation steps are unrealistic in domain adaptation tasks. They also note that these algorithms need to also be evaluated on dataset with significant distribution shift between the two domains, since those are the cases where pseudo-labeling techniques are especially vulnerable.

3 Pseudo-labeling for Classification Enhancement

3.1 Theoretical background

Let $D_{source} = \{x_i^s, y_i^s\}, i = 1, 2, \cdots, N_{source}$ be the set containing the training samples from the source dataset, and $D_{target} = \{x_i^t\}, i = 1, 2, \cdots, N_{target}$ the equivalent from the target dataset. In domain adaptation tasks we assume that the distributions of D_{source} and D_{target} differ significantly, thus we can not simply train a classifier on D_{source} and expect satisfactory results on D_{target}.

Pseudo-labeling uses the probability estimates of a model to create labels y_i^t for the target samples $\{x_i^t\} \in D_{target}$. Given the estimate $p(C|x_i^t; \Theta)$, where Θ the model parameters and C the class label, we assume that $y_i^t = argmax_c p(c|x_i^t; \Theta)$. These samples are then added to the training dataset alongside D_{source} samples. The value $max_c p(c|x_i^t; \Theta)$ is often called the `classification confidence score` of the sample.

This strategy appears easy and intuitive but holds major caveats. If we applied the following algorithm on the entire D_{target} dataset, it is certain that the training would fail because of mislabeled samples, since, had the model's pseudo-predictions been mostly correct, there would be no incentive for the domain adaptation task in the first place. Thus, most such strategies employ an iterative approach, where only samples with a high certainty of correct classification are selected for training. This certainty is usually defined either as a high enough classification score using the same model, or via majority voting by a selection of similar models.

3.2 Model-agnostic Pseudo-labeling algorithm

We use a modified version of the iCAN algorithm [11]. The original iCAN algorithm was designed around a model created with the explicit purpose of learning domain-invariant features between the source and target datasets (CAN model). When the unmodified algorithm uses a generic deep model however a number of issues arise. Our approach seeks to remedy the most major of those issues.

First of all, in the original paper, the authors propose adding the loss of the fully supervised dataset $\mathcal{L}_{source}$, the loss of the target dataset $\mathcal{L}_{tar}$ and the loss of their model's domain-agnostic penalty $\mathcal{L}_{CAN}$ for each mini-batch as $\mathcal{L} = \mathcal{L}_{source} + \mathcal{L}_{tar} + \mathcal{L}_{CAN}$. This design decision most likely exists because of the need to have both a source and a target mini batch loaded on the network in order to calculate $\mathcal{L}_{CAN}$. Since this penalty does not exist in our algorithm, we instead split the training epoch into distinct backward passes for the source and target mini-batches, in order to reduce GPU VRAM requirements.

Secondly, the original iCAN algorithm selects pseudo-labeled batches for each backward pass because of the aforementioned mini-batch requirements. Since our algorithm proves much more unstable, as the underlying generic model may not learn domain-agnostic features, we instead perform pseudo-labeling once every N_{period} epochs. This mechanism ensures that our model will have acquired knowledge of the target distribution from the previously selected pseudo-labeled samples, before being asked to perform pseudo-labeling on the less-easily classifiable, unlabeled samples.

Thirdly, we do not use the "domain bias reweighing function" used by the original authors when calculating $\mathcal{L}_{tar}$. Aside from necessitating a second "domain" classifier, the sampling strategy we employ is inverse to the one proposed by the authors. iCAN attempts to select samples that do not fit the source distribution, in order to prevent its model from selecting target samples that are very similar to the source dataset (since they would score higher confidence scores). Our model-agnostic algorithm attempts to select samples that are closer to the source distribution and, as the model becomes more accustomed to the target distribution, slowly include samples closer to the latter. This is also the motivation behind not re-labeling pseudo-labeled samples.

The modified procedure can be found in Algorithm 1, where $\%$ is the modulo operator. The `adaptive_threshold` function is defined in Section 3.3.

3.3 Adaptive threshold

We use the same `adaptive threshold` function used in the original paper, which adjusts the confidence threshold used to decide whether to pseudo-label a sample. The function is defined as $adaptive\_threshold(acc, \rho) = \frac{1}{1+e^{-\rho * acc}}$, where acc is the accuracy of the classifier, and ρ a tunable hyperparameter, with $\rho = 3$ in the original paper. We evaluate the classifier accuracy on the validation set of the source dataset.

Higher ρ values lead to a steeper decision curve, as see in Figure. The ρ parameter is thus a very convenient way of tuning the pseudo-labeling procedure; a high ρ value leads to samples chosen conservatively, ensuring that more samples are correctly labeled, while lower values lead to more samples overall being chosen, but with more incorrect labels. Since our strategy does not involve re-labeling of pseudo-labeled samples, and because of the lack of external mechanisms to detect outliers, we recommend that $\rho \in [3, 4]$, although the value is dependent on the datasets and underlying classifier.

Algorithm 1 Modified general incremental learning algorithm

```
 1: Train model on dataset $D_{source}$
 2: $D_{pseudo}$ = {}
 3: for epoch do
 4:     if $epoch\%N_{period} = 0$ then
 5:         Calculate $accuracy$ on the validation source dataset
 6:         $\mathcal{T} = adaptive\_threshold(accuracy, \rho)$
 7:         for each $d \in D_{target}$ do
 8:             $label, confidence = model(d)$
 9:             if $confidence > \mathcal{T}$ then
10:                 $D_{pseudo} = D_{pseudo} \cup \{d : label\}$
11:                 $D_{target} = D_{target} - \{d\}$
12:             end if
13:         end for
14:     end if
15:     $D_{rand\_source}$ = {}
16:     Select random samples from $D_{source}$ and add to $D_{rand\_source}$ such as $|D_{rand\_source}| = |D_{pseudo}|$
17:     Train epoch on $D_{rand\_source}$
18:     Train epoch on $D_{pseudo}$
19: end for
```

Overall, the adaptive threshold function provides us with an easy-to-understand mechanism which both adapts to the current state of the model, and which can be tuned with a single hyperparameter. This hyperparameter however remains of crucial importance, and should be tuned either by the number of accepted samples, or if available, a target validation dataset which tracks the pseudo-labeling misclassifications.

3.4 Semi-supervised learning for learning domain-invariant features

One of the greatest challenges in the unsupervised pseudo-labeling approach is the distance between the source and target distributions, which may inhibit training. By using Algorithm 1 during experimentation we noticed three dominant patterns in the evolution of the model:

- The model selects samples liberally from D_{target}. As established in Section 3.3, this usually leads to a complete de-volution in training as the model makes increasingly more incorrect decisions, as it mislabels its own input.

- The model selects samples conservatively from a few classes. In this case, the model correctly attributes samples from a few easily distinguishable classes in D_{target} (”calculator” and ”keyboard” in our experiments). It then assumes that all samples bearing features from D_{target} belong in these classes and begins mislabeling all other classes.

- The model selects samples much more conservatively but using most classes. This usually leads to a hyper-conservative model which selects samples only close to the original source distribution, and ends up disregarding a large portion of D_{target}.

In all three cases the underlying problem is that we can not, by design, expect a generic model to learn domain-invariant features.

A very simple way to partially overcome this problem is to label a small portion of D_{target} and include it during the initial pretraining phase. In other words, we pretrain the model on $\tilde{D}_{source} = D_{source} \cup D_{labeled\_target}$, instead of D_{source}, where $D_{labeled\_target}$ is a stratified, labeled sample of D_{target}. This allows the model to bypass the conditions for the class imbalance problem described above, and may provide the opportunity for our model to more pick many additional samples during the first few rounds of pseudo-labeling bypassing the hyper-conservative model issue.

	MNIST/MNIST-M	Modern Office-31
#Samples (source)	70,000	2,817
#Samples (target)	149,002	795
#Classes	10	31
Task Difficulty	Easy/Medium	High

Table 1: Dataset characteristics for MNIST/MNIST-M and Modern Office-31

4 Experiments

4.1 Datasets

In this paper we examine the effects of ”vanilla” pseudo-labeling on two main domain adaptation datasets focused on image classification.

The first domain adaptation task we explore is the ”MNIST to MNIST-M” task. The MNIST [6] dataset is a widely-used benchmark in machine learning, particularly for handwritten digit recognition. It consists of 70,000 grayscale images of handwritten digits (0-9), split into 60,000 training images and 10,000 test images, each sized 28x28 pixels. MNIST-M [3] is a variant of the MNIST dataset created to present a more challenging domain adaptation problem. It overlays the original MNIST digits onto random background images extracted from color photos, adding variability and noise to the digit images. MNIST-M retains the same structure and digit classes as MNIST but introduces complex and colorful backgrounds to the 28x28 images.

The second dataset is the Modern Office-31 [9] dataset, a newly introduced dataset designed to serve as a drop-in replacement for the well-known Office-31 [10] dataset. It maintains the same 31 classes as Office-31 (computers, keyboards, and office furniture, captured in diverse settings and under different conditions), while also including real-life domain images from the original Office-31 webcam images and a cleaned Amazon domain, creating a comprehensive dataset with 7,210 images across three domains: Amazon, Synthetic, and Webcam. In this study, we use the ”Amazon to Webcam” task to evaluate our model.

Thus for our experiments, we can compare results with a task where the source and target distributions are fairly close (MNIST to MNIST-M), and one where they are significantly diverse from each other (Modern Office Amazon to Webcam). We hope to address in this way, the common criticism of evaluating domain adaptation tasks on tasks with similar domains, as noted by [8]. A comparison of these datasets can be found in Table 1.

4.2 Experimental Setup

For each domain adaptation task we produce two models, one trained on the source dataset and one on the target dataset, as baselines. These models serve as the lower and upper bounds of the classifiers' performance respectively.

We then finetune three pretrained models on different source datasets; D_{source} (unsupervised), $\tilde{D}_{source_{10}} = D_{source} \cup D_{labeled\_target_{10}}$ and $\tilde{D}_{source_{20}} = D_{source} \cup D_{labeled\_target_{20}}$ (semi-supervised), where $D_{labeled\_target_{\chi}}$ is a random, labeled sample from the target dataset, with $\chi\%$ samples. The finetuned models are then used as input for the incremental learning algorithm outlined in Section 3.2.

We monitor two metrics for each of these models; the performance on the target test set, and the rate of pseudo-labeling misclassifications. Of course, the latter practically necessitates the a-priori knowledge of the pseudo-labeled samples' labels. We thus use this metric **only** as a post-analysis metric to demonstrate the efficiency of pseudo-labeling for each model and dataset, and **not as a validation metric** to tune the model's training. Our experience does suggest however that performing this procedure on a validation set can significantly help in both tuning hyperparameters, and diagnosing issues during training.

We use the ResNet-18 [5] model for all experiments trained with the Adam optimizer ($lr = 0.0005$). For the MNIST/MNIST-M dataset in particular, we use the PyTorch AdamW implementation with $weight\_decay = 10^{-3}$. We use different label smoothing parameters for the finetuning and incremental learning phases. We set $\alpha = 0.05$ for the finetuning task, and $\alpha = 0.15$ for the incremental learning, in order to account for the greater possibility of pseudo-labeled misclassifications. We set the adaptive threshold rate for the MNIST/MNIST-M task $\rho = 3$, as in its original paper [11], and $\rho = 4$ for the Modern Office-31 task, in order to account for the larger shift in distributions. Each ResNet model is pretrained on the ImageNet [1] dataset, before being

	MNIST/MNIST-M	**Modern Office-31**
Source-only	33%	12%
Target-only	97%	Untrainable
Unsupervised	37%	25%
Supervised 10%	91%	23%
Supervised 10% + pseudo-labeling	94%	16%
Supervised 20%	93%	34%
Supervised 20% + pseudo-labeling	93%	33%

Table 2: Comparison of different methods on MNIST/MNIST-M and ModernOffice-31

finetuned on our respective source datasets. Datasets for both domains are randomly split in a 70%-15%-15% training-validation-test split.

4.3 Results

Table 2 presents the performance of various domain adaptation methods on the MNIST/MNIST-M and ModernOffice-31 datasets, highlighting the effectiveness of different training approaches. For the MNIST/MNIST-M dataset, the Source-only method yields a low accuracy of 33%, underscoring the challenges of domain shift. In contrast, the Target-only method achieves a high accuracy of 97%, illustrating the upper bound of performance when extensive labeled target data is available. Unsupervised adaptation slightly improves the accuracy to 37%, demonstrating some benefit but indicating the limitations of purely unsupervised approaches. Supervised methods with 10% labeled target data achieve high accuracies of 91% without incremental learning and 94% with incremental learning, showing the same rate of improvement with the source-only model. Similarly, using 20% labeled target data yields accuracies of 93% for both incremental and non-incremental approaches, indicating that when extensive labeled data are present, pseudo-labeling does not yield any improvements in efficiency.

For the Modern Office-31 dataset, the Source-only method achieves a mere 12% accuracy, again highlighting domain adaptation challenges. The Target-only method is marked as untrainable, due to the very limited number of target samples. Unsupervised adaptation improves accuracy to 25%, suggesting some domain alignment. Supervised approaches with 10% labeled target data achieve 23% accuracy without incremental learning and 16% with it, suggesting that incremental learning may not always be beneficial. With 20% labeled target data, the accuracies are 34% and 33% for non-incremental and incremental learning, respectively, indicating a notable improvement over unsupervised methods.

References

[1] Jia Deng et al. 'ImageNet: A large-scale hierarchical image database'. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[2] Fabian Dubourvieux et al. 'A formal approach to good practices in Pseudo-Labeling for Unsupervised Domain Adaptive Re-Identification'. In: (2022). arXiv: 2112.12887 [cs.CV].

[3] Yaroslav Ganin et al. *Domain-Adversarial Training of Neural Networks*. 2016. arXiv: 1505.07818 [stat.ML].

[4] Yves Grandvalet and Yoshua Bengio. 'Semi-supervised Learning by Entropy Minimization'. In: *Conférence francophone sur l'apprentissage automatique*. 2004. URL: https://api.semanticscholar.org/CorpusID:7890982.

[5] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[6] Y. Lecun et al. 'Gradient-based learning applied to document recognition'. In: *Proceedings of the IEEE* 86.11 (Nov. 1998), pp. 2278–2324. ISSN: 1558-2256. DOI: 10.1109/5.726791.

[7] Dong-Hyun Lee. 'Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks'. In: *ICML 2013 Workshop : Challenges in Representation Learning (WREPL)* (July 2013).

[8] Avital Oliver et al. ‘Realistic Evaluation of Semi-Supervised Learning Algorithms’. In: 2018. URL: https://arxiv.org/pdf/1804.09170.pdf.

[9] Tobias Ringwald and Rainer Stiefelhagen. ‘Adaptiope: A Modern Benchmark for Unsupervised Domain Adaptation’. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV).* Jan. 2021, pp. 101–110.

[10] Kate Saenko et al. ‘Adapting Visual Category Models to New Domains’. In: *Computer Vision – ECCV 2010.* Ed. by Kostas Daniilidis, Petros Maragos and Nikos Paragios. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–226. ISBN: 978-3-642-15561-1.

[11] Weichen Zhang et al. ‘Collaborative and Adversarial Network for Unsupervised Domain Adaptation’. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2018, pp. 3801–3809. DOI: 10.1109/CVPR.2018.00400.

[12] Youshan Zhang. *A Survey of Unsupervised Domain Adaptation for Visual Recognition.* 2021. arXiv: 2112.06745 [cs.CV].